

EAST Search History

Ref #	Hits	Search Query	DBs	Default Operator	Plurals	Time Stamp
L1	396	717/130.ccls.	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2006/12/21 13:47
L2	421	717/131.ccls.	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2006/12/21 13:48
L3	187	717/153.ccls.	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2006/12/21 13:48
L4	35	717/153.ccls. and (hot or frequent\$2 of frequenc\$3) and (instrument\$5 or patch\$3 hook\$3)	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2006/12/21 13:50
L5	137	717/130.ccls. and (hot or frequent\$2 of frequenc\$3) and (instrument\$5 or patch\$3 hook\$3)	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2006/12/21 13:50
L6	76	717/131.ccls. and (hot or frequent\$2 of frequenc\$3) and (instrument\$5 or patch\$3 hook\$3)	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2006/12/21 13:51
L7	415	stor\$3 near\$3 instrument\$5 and (redirect\$3 or divert\$3)	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2006/12/21 13:51
L8	1	I4 and I7	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2006/12/21 13:52
L9	12	I5 and I7	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2006/12/21 13:52

EAST Search History

L10	5	I6 and I7	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2006/12/21 13:52
L11	14	I8 I9 I10	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2006/12/21 13:52
S1	353	717/130.ccls.	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2006/06/08 07:37
S2	381	717/131.ccls.	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2006/06/08 07:37
S3	167	717/153.ccls.	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2006/06/08 07:38
S4	1098	((profil\$3 or instrument\$5) same (frequent\$2 or frequency or hot)) and (policy or rule or threshold or limit or count\$3 or "number of") and (fetch\$3 or ("before" adj (pipelin\$3 or execut\$3))) and (log\$4 or collect\$3 or record\$3 or document\$5) and (executable or "machine code" or binary or binaries or compiled)	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2006/06/08 09:08
S5	171	717/???ccls. and S4	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2006/06/08 07:46
S6	1490	instrument\$5 near2 (frequently or frequent or hot)	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2006/06/08 08:57
S7	511	instrument\$5 adj (frequently or frequent or hot)	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2006/06/08 08:46

EAST Search History

S8	7	instrument\$5 adj (frequently or frequent or hot) and 717/???.ccls.	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2006/06/08 07:57
S9	2	"6470492".pn.	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2006/06/08 07:57
S10	20	("3707725" "5751982" "6148437" "6189141" "6205545" "6219827" "6237065").PN. OR ("6470492").URPN.	US-PGPUB; USPAT; USOCR	OR	OFF	2006/06/08 07:58
S11	20	("3707725" "5751982" "6148437" "6189141" "6205545" "6219827" "6237065").PN. OR ("6470492").URPN. and (frequently or frequent or frequency or hot or "number of" or count\$3)	US-PGPUB; USPAT; USOCR	OR	OFF	2006/06/08 08:00
S12	0	instrument\$5 adj (frequently or frequent or hot) and (path near3 profil\$3)	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2006/06/08 08:47
S13	2	instrument\$5 adj (frequently or frequent or hot) and (caller or callee or (call adj (tree or stack)) near3 profil\$3)	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2006/06/08 08:52
S14	2	instrument\$5 adj (frequently or frequent or hot) and (caller or callee or (call adj (tree or stack)))	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2006/06/08 08:52
S15	475	"profiling tool"	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2006/06/08 08:57
S16	2	S6 and S15	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2006/06/08 08:58

EAST Search History

S17	54	("4071744" "5133072" "5193180" "5212794" "5335344" "5355491" "5367685" "5369766" "5381534" "5414855" "5442790" "5450586" "5452457" "5487158" "5504914" "5519866" "5522036" "5522072" "5535329" "5548794" "5581697" "5583988" "5590331" "5606697" "5625832" "5627981" "5628016" "5644742" "5652884" "5655121" "5713010" "5732210" "5751985" "5764962" "5778245" "5815720" "5838978" "5854928" "5857103" "5909578" "5911073" "5933622" "6052530").PN. OR ("6164841").URPN.	US-PGPUB; USPAT; USOCR	OR	OFF	2006/06/08 09:08
S18	54	(profil\$3 or instrument\$5 or frequent\$2 or frequency or hot or policy or rule or threshold or limit or count\$3 or "number of" or fetch\$3 or ("before" adj (pipelin\$3 or execut\$3)) or log\$4 or collect\$3 or record\$3 or document\$5 or executable or "machine code" or binary or binaries or compiled or call\$3) and S17	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2006/06/12 12:53
S19	4	"20040103401" "6662359".pn.	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2006/06/08 09:32
S20	23	("5974549" "6275938" "5768593").pn. "20030093650" "20030101292" "20030101330" "20030101334" "2030101381" "20030101431" "20030101439" "20030182653" "20030192035"	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2006/06/08 09:42
S21	25	("5974549" "6275938" "5768593").pn. "20030093650" "20030101292" "20030101330" "20030101334" "20030101381" "20030101431" "20030101439" "20030182653" "20030192035"	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2006/06/08 09:43
S22	3408	(profil\$3 or instrument\$5 or frequent\$2 or frequency or hot or policy or rule or threshold or limit or count\$3 or "number of" or fetch\$3 or ("before" adj (pipelin\$3 or execut\$3)) or log\$4 or collect\$3 or record\$3 or document\$5 or executable or "machine code" or binary or binaries or compiled or call\$3) and vliw	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2006/06/12 12:47

EAST Search History

S23	2	(profil\$3 or intercept\$3) near5 (cach\$3 or journal\$3 or stor\$3) and ("number of" or count\$3 or frequency or hot) near5 vliw and cach\$3 and (instrument\$5 or insert\$3 or add\$3)	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2006/06/12 13:00
S24	52	("number of" or count\$3 or frequency or hot) near5 vliw and cach\$3 and (instrument\$5 or insert\$3 or add\$3)	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2006/06/12 13:02
S25	2	("number of" or count\$3 or frequency or hot) near5 vliw and cach\$3 and (instrument\$5 or insert\$3 or add\$3) and 717/???.cccls.	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2006/06/12 13:01
S26	34	("number of" or count\$3 or frequency or hot) near3 vliw and cach\$3 and (instrument\$5 or insert\$3 or add\$3)	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2006/06/12 13:05
S27	39	("number of" or count\$3 or frequency or hot) near3 (vliw or "IA 64" or "C6 DSP" or epic) and cach\$3 and (instrument\$5 or insert\$3 or add\$3)	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2006/06/12 13:13
S28	0	(execut\$3 near3 ("number of" or count\$3 or frequency or hot)) near5 (vliw or "IA 64" or "C6 DSP" or epic) and cach\$3 and (instrument\$5 or insert\$3 or hook\$3)	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2006/06/12 13:15
S29	3	(execut\$3 near3 ("number of" or count\$3 or frequency or hot)) near5 (vliw or "IA 64" or "C6 DSP" or epic)	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2006/06/12 13:16
S30	19	(instruction near3 ("number of" or count\$3 or frequency or hot)) near5 (vliw or "IA 64" or "C6 DSP" or epic)	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2006/06/12 13:18
S31	28	(instruction near3 ("number of" or count\$3 or frequency or hot)) near5 (vliw or "IA 64" or "C6 DSP" or epic or superscalar or "super scalar")	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2006/06/12 13:19
S32	5	(instruction near3 ("number of" or count\$3 or frequency or hot)) near5 (vliw or "IA 64" or "C6 DSP" or epic or superscalar or "super scalar") and (trace or tracing or profil\$3)	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2006/06/12 13:32

EAST Search History

S33	1	(instruction near3 ("number of" or count\$3 or frequency or hot)) near5 (vliw or "IA 64" or "C6 DSP" or epic or superscalar or "super scalar") and (trace or tracing or profil\$3) and frequency and "6178492".pn.	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2006/06/12 13:33
-----	---	--	---	----	----	------------------


[Subscribe \(Full Service\)](#) [Register \(Limited Ser](#)
[Search: The ACM Digital Library](#) [The](#)
[+instrument +hot +instructions](#)
[Feedback](#) [Report a problem](#)

Published since January 1985 and Published before June 2003

Terms used **instrument** **hot** **instructions**

Sort results by [Save results to a Binder](#) [Try an Advanced](#)
 Display results [Search Tips](#) [Try this search](#)
☐ Open results in a new window

Results 1 - 20 of 200

Result page: [1](#) [2](#) [3](#) [4](#) [5](#) [6](#) [7](#) [8](#) [9](#) [10](#) [next](#)

Best 200 shown

Re

1 [Optimising hot paths in a dynamic binary translator](#)

David Ung, Cristina Cifuentes

March 2001 **ACM SIGARCH Computer Architecture News**, Volume 2

Publisher: ACM Press

Full text available: [pdf\(890.10 KB\)](#) Additional Information: [full citation](#), [abstract terms](#)

In dynamic binary translation, code is translated "on the fly" at run-time, perceives ordinary execution of the program on the target machine. Code frequently executed follow the same sequence of flow control over a per fragments form a hot path and are optimised to improve the overall performance. Multiple hot paths may also exist in programs. A program may one hot path for some time, but later switch to another ...

Keywords: binary translation, dynamic compilation, dynamic execution


2 [Efficient instrumentation for code coverage testing](#)

Mustafa M. Tikir, Jeffrey K. Hollingsworth

July 2002 **ACM SIGSOFT Software Engineering Notes**, Proceedings of SIGSOFT international symposium on Software testing and

'02, Volume 27 Issue 4


Publisher: ACM Press

Full text available:  [pdf\(524.54 KB\)](#) Additional Information: [full citation](#), [abstracts](#)

Evaluation of Code Coverage is the problem of identifying the parts of a program that do not execute in one or more runs of a program. The traditional approach for tools is to use static code instrumentation. In this paper we present a new technique to dynamically insert and remove instrumentation code to reduce the runtime overhead. We also explore the use of dominator tree information to reduce the number of instrumentation points needed. Our experiments show that ...


Keywords: code coverage, dominator tree, dynamic code deletion, dynamic on-demand instrumentation, testing

3 Dynamo: a transparent dynamic optimization system

 Vasanth Bala, Evelyn Duesterwald, Sanjeev Banerjia

May 2000 **ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN Conference on Programming language design and implementation PLDI 2000**
Issue 5

Publisher: ACM Press

Full text available:  [pdf\(156.03 KB\)](#) Additional Information: [full citation](#), [abstracts](#), [index terms](#)

We describe the design and implementation of Dynamo, a software dynamic optimization system that is capable of transparently improving the performance of a native instruction stream as it executes on the processor. The input native instruction stream can be dynamically generated (by a JIT for example), or it can come from the execution of a statically compiled native binary. This paper evaluates the Dynamo system in a more challenging situation, in order to emphasize the ...

4 Hot cold optimization of large Windows/NT applications

Robert Cohn, P. Geoffrey Lowney

December 1996 **Proceedings of the 29th annual ACM/IEEE international symposium on Microarchitecture**

Publisher: IEEE Computer Society

Full text available:  [pdf\(1.14 KB\)](#) Additional Information: [full citation](#), [abstracts](#)

MB)citings, index ter

A dynamic instruction trace often contains many unnecessary instruction only by the unexecuted portion of the program. Hot-cold optimization (HCO) realizes this performance opportunity. HCO uses profile information to partition the routine into frequently executed (hot) and infrequently executed (cold) portions. Operations in the hot portion are removed, and compensation code is added from hot to cold as needed. We evaluate HCO on a ...


Keywords: optimization, profile, NT, register allocation

5 Efficient and flexible value sampling

◆ M. Burrows, U. Erlingsson, S.-T. A. Leung, M. T. Vandevoorde, C. A. Walker, W. E. Weihl

November 2000 **ACM SIGPLAN Notices**, Volume 35 Issue 11

Publisher: ACM Press

Full text available:  [pdf\(973.28 KB\)](#) Additional Information: [full citation](#), [abstract](#), [citings](#), [index terms](#)


This paper presents novel sampling-based techniques for collecting statistics on register contents, data values, and other information associated with instructions. Memory latencies. Values of interest are sampled in response to periodic interrupts. Resulting value profiles can be analyzed by programmers and optimizers to improve the performance of production uniprocessor and multiprocessor systems. Our system extends the DCPI continuous profiling infrastructure ...

6 Efficient and flexible value sampling

◆ M. Burrows, U. Erlingsson, S.-T. A. Leung, M. T. Vandevoorde, C. A. Walker, W. E. Weihl

November 2000 **ACM SIGOPS Operating Systems Review**, **ACM SIGARCH Architecture News**, **Proceedings of the ninth international conference on Architectural support for programming languages and operating systems (ASPLOS-IX)**, Volume 34, Issue 5, 5

Publisher: ACM Press

Full text available:  [pdf\(191.88 KB\)](#) Additional Information: [full citation](#), [abstract](#), [citings](#), [index terms](#)

This paper presents novel sampling-based techniques for collecting statistics on register contents, data values, and other information associated with instructions. Memory latencies. Values of interest are sampled in response to periodic interrupts. Resulting value profiles can be analyzed by programmers and optimizers to improve the performance of production uniprocessor and multiprocessor systems. Our system extends the DCPI continuous profiling infrastructure ...

register contents, data values, and other information associated with instruction memory latencies. Values of interest are sampled in response to periodic resulting value profiles can be analyzed by programmers and optimizers performance of production uniprocessor and multiprocessor systems. Our system extends the DCPI continuous profiling infrastructure ...

7 Exploiting hardware performance counters with flow and context sensitive



Glenn Ammons, Thomas Ball, James R. Larus

May 1997 **ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN on Programming language design and implementation PLDI**
Issue 5

Publisher: ACM Press

Full text available: [pdf\(1.67 MB\)](#) Additional Information: [full citation](#), [abstracts](#), [citations](#), [index terms](#)

A program profile attributes run-time costs to portions of a program's execution. Profiling systems suffer from two major deficiencies: first, they only approximate metrics, such as execution frequency or elapsed time to static, syntactic constructs or procedures or statements; second, they aggressively reduce the volume of data collected and reported, although aggregation can hide striking differences in behavior. This paper addresses both concerns by exploiting the hardware ...

8 A framework for reducing the cost of instrumented code



Matthew Arnold, Barbara G. Ryder

May 2001 **ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN on Programming language design and implementation PLDI**
Issue 5

Publisher: ACM Press


Full text available: [pdf\(1.78 MB\)](#) Additional Information: [full citation](#), [abstracts](#), [citations](#), [index terms](#)

Instrumenting code to collect profiling information can cause substantial overhead. This overhead makes instrumentation difficult to perform at run time, preventing many known *offline* feedback-directed optimizations from being applied to systems. This paper presents a general framework for performing *instrumentation* to reduce the overhead of previously expensive instrumentation. The framework is fast and effective, using code-duplication and *countdown* ...

9 A dynamic optimization framework for a Java just-in-time compiler

- ◆ Toshio Suganuma, Toshiaki Yasue, Motohiro Kawahito, Hideaki Komatsu
October 2001 **ACM SIGPLAN Notices , Proceedings of the 16th ACM
conference on Object oriented programming, systems, languages
applications OOPSLA '01**, Volume 36 Issue 11

Publisher: ACM Press


Full text available:  [pdf\(2.12 MB\)](#) Additional Information: [full citation](#), [abstracts](#), [citations](#), [index terms](#)

The high performance implementation of Java Virtual Machines (JVM) : (JIT) compilers is directed toward adaptive compilation optimizations or runtime profile information. This paper describes the design and implementation of a dynamic optimization framework in a production-level Java JIT compiler to employ a mixed mode interpreter and a three level optimizing compiler: full, and special optimization, each of which has a different ...

10 Profile-based optimizations: Dynamic trace selection using performance measurement sampling

- Howard Chen, Wei-Chung Hsu, Jiwei Lu, Pen-Chung Yew, Dong-Yuan Chao
March 2003 **Proceedings of the international symposium on Code generation and optimization: feedback-directed and runtime optimization**

Publisher: IEEE Computer Society


Full text available:  [pdf\(1.88 MB\)](#) Additional Information: [full citation](#), [abstracts](#), [citations](#), [index terms](#)

Optimizing programs at run-time provides opportunities to apply aggressive optimizations to programs based on information that was not available at compile time. Adaptive programs can be adapted to better exploit architectural features, optimize code, and simplify code based on run-time constants. Our profiling system is a framework for collecting information required for performing run-time code optimization. We sample the performance hardware registers available on ...

11 Dynamic hot data stream prefetching for general-purpose programs

- ◆ Trishul M. Chilimbi, Martin Hirzel
May 2002 **ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN
on Programming language design and implementation PLDI
Issue 5**

Publisher: ACM Press

Full text available:  [pdf\(210.85 KB\)](#) Additional Information: [full citation](#), [abstracts](#), [citations](#), [index terms](#)


Prefetching data ahead of use has the potential to tolerate the growing performance gap by overlapping long latency memory accesses with use. While sophisticated prefetching techniques have been automated for limited scientific codes that access dense arrays in loop nests, a similar level of prefetching is needed in general-purpose programs, especially pointer-chasing codes written in languages such as C and C++. We address this problem by describing ...


Keywords: data reference profiling, dynamic optimization, dynamic performance optimization, prefetching, temporal profiling

12 Compilation and run-time systems: Vacuum packing: extracting hardware-phases for post-link optimization

Ronald D. Barnes, Erik M. Nystrom, Matthew C. Merten, Wen-mei W. Hwu
November 2002 **Proceedings of the 35th annual ACM/IEEE international symposium on Microarchitecture**

Publisher: IEEE Computer Society Press

Full text available:  [pdf\(1.26 MB\)](#)

 [Publisher Site](#)

Additional Information: [full citation](#), [abstracts](#), [citations](#), [index terms](#)


This paper presents Vacuum Packing, a new approach to profile-based post-link optimization. Instead of using traditional aggregate or summarized execution weights, this approach uses a transparent hardware profiler to automatically extract hardware phases and record branch profile information for each new phase. The compiler algorithm then produces code packages that are specially formed for these phases. The algorithm compensates for the incomplete and often incoherent

13 Dynamic Adaptive compilation: Dynamic profiling and trace cache generation

Marc Berndt, Laurie Hendren

March 2003 **Proceedings of the international symposium on Code generation and compilation: feedback-directed and runtime optimization**

Publisher: IEEE Computer Society

Full text available:  [pdf\(950.33 KB\)](#) Additional Information: [full citation](#), [abstracts](#), [citations](#), [index terms](#)

KB)index terms


Dynamic program optimization is increasingly important for achieving performance. A key issue is how to select which code to optimize. One can dynamically detect traces, long sequences of instructions spanning multiple basic blocks, that are likely to execute to completion. Traces are easy to optimize and have a good unit for optimization. This paper reports on a new approach for dynamically creating and storing traces in a Java virtual machine. We ...

14 Memory aware compilation through accurate timing extraction

◆ Peter Grun, Nikil Dutt, Alex Nicolau

June 2000 **Proceedings of the 37th conference on Design automation**

Publisher: ACM Press

Full text available:  [pdf\(79.24 KB\)](#) Additional Information: [full citation](#), [abstracts](#), [index terms](#)


Memory delays represent a major bottleneck in embedded systems performance. Memory modules exhibiting efficient access modes (e.g., page-, burst-memory access) can exploit this bottleneck. However, such features can not be efficiently exploited in embedded systems without memory-aware compiler support. We describe a compiler approach that exploits such efficient memory access modes by using timing information, allowing the compiler's scheduler to ...

15 Recompilation for debugging support in a JIT-compiler

◆ Mustafa M. Tikir, Jeffrey K. Hollingsworth, Guei-Yuan Lueh

November 2002 **ACM SIGSOFT Software Engineering Notes , Proceedings of the ACM SIGPLAN-SIGSOFT workshop on Program analysis and tools and engineering PASTE '02**, Volume 28 Issue 1

Publisher: ACM Press

Full text available:  [pdf\(89.55 KB\)](#) Additional Information: [full citation](#), [abstracts](#), [index terms](#)

A static Java compiler converts Java source code into a verifiably secure architecture-neutral intermediate format, called Java *byte codes*. The Java byte codes are either interpreted by a Java Virtual Machine or translated into native code by just-in-time compilers. Static Java compilers embed debug information in the Java byte codes used by the source level debuggers. However, the debug information is not architecture independent byte codes and most often ...


Keywords: Java, Java virtual machine debugger interface, debug inform recompilation, field access watch, just-in-time compilation

16 Optimized code restructuring of OS/2 executables

Jyh-Herng Chow, Yong-fong Lee, Kalyan Muthukumar, Vivek Sarkar, Ma Garcia, John Hsu, Shauchi Ong, Honesty Young

November 1995 **Proceedings of the 1995 conference of the Centre for A on Collaborative research**

Publisher: IBM Press

Full text available:  [pdf\(234.83 KB\)](#) Additional Information: [full citation](#), [abst index terms](#)


This paper describes the design and algorithms of FDPR/2 (Feedback Di Restructuring of OS/2 executables), a general-purpose tool that can be us profile, and restructure/optimize OS/2 executables for the tel x86 archite optimizations delivered by FDPR/2's restructuring include improved util (instruction) memory hierarchy, improved branch alignment, and dead c These optimizations are known to be critical for object-oriented pro ...

17 Whole program paths

 James R. Larus

May 1999 **ACM SIGPLAN Notices , Proceedings of the ACM SIGPLA on Programming language design and implementation PLD** Issue 5

Publisher: ACM Press

Full text available:  [pdf\(1.25 MB\)](#) Additional Information: [full citation](#), [abst citings](#), [index ter](#)

Whole program paths (WPP) are a new approach to capturing and repres dynamic---actually executed---control flow. Unlike other path profiling t record intraprocedural or acyclic paths, WPPs produce a single, compact program's entire control flow, including loop iteration and interprocedur explains how to collect and represent WPPs. It also shows how to use W *subpaths*, which are the heavily executed ...


Keywords: data compression, dynamic program measurement, path prof control flow, program tracing

18 Improving data-flow analysis with path profiles

◆ Glenn Ammons, James R. Larus

May 1998 **ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN on Programming language design and implementation PLDI**
Issue 5

Publisher: ACM Press

Full text available:  [pdf\(1.57 MB\)](#) Additional Information: [full citation](#), [abstracts](#), [citations](#), [index terms](#)


Data-flow analysis computes its solutions over the paths in a control-flow graph---whether feasible or infeasible, heavily or rarely executed---control flow paths. However, programs execute only a small fraction of their potential paths. Moreover, programs' execution time and cost is concentrated in a few paths. This paper describes a new approach to analyzing and optimizing programs that improves the precision of data flow analysis at ...

19 Non-intrusive and interactive profiling in parasight

◆ Ziya Aral, Ilya Gertner

January 1988 **ACM SIGPLAN Notices , Proceedings of the ACM/SIGPLAN Parallel programming: experience with applications, languages, and tools PPEALS '88**, Volume 23 Issue 9

Publisher: ACM Press

Full text available:  [pdf\(1.05 MB\)](#) Additional Information: [full citation](#), [abstracts](#), [citations](#), [index terms](#)

Debugging the performance of parallel applications is crucial for fully utilizing multiprocessor hardware. This paper describes profiling tools in the Parasight programming environment that is geared towards non-intrusive performance debugging. In Parasight, profilers execute as observer programs concurrently with the target program and monitor its behavior. This design provides experience in debugging and monitoring the performance of ...

20 Characterizing memory hot spots in a shared memory MIMD machine

◆ Raymond R. Glenn, Daniel V. Pryor, John M. Conroy, Theodore Johnson

August 1991 **Proceedings of the 1991 ACM/IEEE conference on Supercomputing**

Publisher: ACM Press

Full text available:  [pdf\(1.18 MB\)](#) Additional Information: [full citation](#), [references](#)

MB)

Results 1 - 20 of 200

Result page: [1](#) [2](#) [3](#) [4](#) [5](#) [6](#) [7](#) [8](#) [9](#) [10](#)

The ACM Portal is published by the Association for Computing Machinery
ACM, Inc.

[Terms of Usage](#) [Privacy Policy](#) [Code of Ethics](#) [Contact](#)

Useful downloads: [!\[\]\(17acf1afa8cdf0b67c53d4865a5ed469_img.jpg\) Adobe Acrobat](#) [!\[\]\(ece8cabb5adcd402275b8866019cc3b8_img.jpg\) QuickTime](#) [!\[\]\(4fe6c1f6e7bbe5a2699a4abd6267bb58_img.jpg\) Windows Med
Player](#)

[Home](#) | [Login](#) | [Logout](#)**IEEE Xplore**
RELEASE 2.1**Welcome United States Patent and
Trademark Office**□ **Search Results****BROWSE SEARCH** **IEEE
GUIDE**

Results for "(((instrument* hot cache reduc* overhead)<in>metadata))
>= 1985 <and>...
Your search matched 0 documents.
A maximum of 100 results are displayed, 25 to a page, sorted by Relevance
Descending order.

» **Search Options**[View Session
History](#)[New Search](#)**Modify Search**(((instrument* hot cache reduc* overhead)<in>metar☐ Check to search only within this results set» **Key****IEEE
JNL** IEEE
Journal or
Magazine**IEEE
JNL** IEEE Journal
or Magazine**IEEE
CNF** IEEE
Conference
Proceeding**IEEE
CNF** IEE
Conference
Proceeding**IEEE
STD** IEEE
Standard**Display
Format:** ☒ Citation ☐ Citation &
Abstract**No results were found.**Please edit your search criteria and try again. Refer
assistance revising your search.Indexed by
 Inspec

[Home](#) | [Login](#) | [Logout](#)**IEEE Xplore**
RELEASE 2.1**Welcome United States Patent and
Trademark Office**□ **Search Results****BROWSE SEARCH IEEE
GUIDE**

Results for "(((instrument* frequen* reduc* overhead)<in>metadata))
>= 1985 <and>...
Your search matched 0 documents.
A maximum of 100 results are displayed, 25 to a page, sorted by **Relevance**
Descending order.

» **Search Options**[View Session
History](#)[New Search](#)**Modify Search**/(((instrument* frequen* reduc* overhead)<in>metadata☐ Check to search only within this results set» **Key****IEEE
JNL**IEEE
Journal or
Magazine**IEE
JNL**IEE Journal
or Magazine**IEEE
CNF**IEEE
Conference
Proceeding**IEE
CNF**IEE
Conference
Proceeding**IEEE
STD**IEEE
Standard**Display** ☒ Citation ☐ Citation &
Format: Abstract**No results were found.**Please edit your search criteria and try again. Refer
assistance revising your search.Indexed by
 Inspec



instrument hot cache reduce profiling

1985

- 2

Scholar [All articles](#) [Recent articles](#) Results 1 - 10 of about 379 for **instrume**

All Results Did you mean: instrument *host* cache reduce profiling

J Larus A framework for reducing the cost of instrumented code - group of 15 »

T Chilimbi M Arnold, BG Ryder - ACM SIGPLAN Notices, 2001 - portal.acm.org

G Ammons ... specific, and may introduce complexities such as maintaining **cache** consistency ... Finally, an adaptive system will likely **instrument** only the **hot** methods, and ...

M Arnold Cited by 110 - [Related Articles](#) - [Web Search](#) - [BL Direct](#)

A Maynard Exploiting hardware performance counters with flow and context sensitive **profiling** - group of 11 »

G Ammons, T Ball, JR Larus - ACM SIGPLAN Notices, 1997 - portal.acm.org

... Tools that report **cache** misses at the procedure or statement level cannot isolate these **hot** paths. ... This paper describes how to **instrument** programs to record ...

Cited by 151 - [Related Articles](#) - [Web Search](#) - [BL Direct](#)

Dynamic **hot** data stream prefetching for general-purpose programs - group of 14 »

TM Chilimbi, M Hirzel - Proceedings of the ACM SIGPLAN 2002 Conference on ..., 2002 - portal.acm.org

... of program references and more than 80% of **cache** misses [8 ... trace, the subsequence abcde is a **hot** data stream ... A common way to **reduce** the overhead of **profiling** is ...

Cited by 67 - [Related Articles](#) - [Web Search](#) - [BL Direct](#)

Bursty tracing: A framework for low-overhead temporal profiling - group of 10 »

M Hirzel, T Chilimbi - 4th ACM Workshop on Feedback-Directed and Dynamic ..., 2001 - cs.colorado.edu

... multiple-exit code regions [6, 13]; **cache**-conscious data ...

We do not **instrument** either

version of the code of ... Figure 7: Overlap of **hot** data streams when sampling ...

Cited by 33 - Related Articles - View as HTML - Web Search

Reducing **cache** misses using hardware and software page placement - group of 13 »

T Sherwood, B Calder, J Emer - Proceedings of the 13th international conference on ..., 1999 - portal.acm.org

... f 0). We used ATOM [3 I] to **instrument** the programs ...

The conflict detection mechanism

looks for **hot** sets as ... section 2. As described earlier, the **cache** sets are ...

Cited by 21 - Related Articles - Web Search

Optimizing Alpha Executables on Windows NT with Spike - group of 11 »

R Cohn, D Goodwin, PG Lowney... - Digital Technical Journal, 1997 - research.compaq.com

... of counters required to fully **instrument** an image. ... system uses code layout and

hot-cold optimization to ... in memory, thereby **reducing** instruction **cache** miss and ...

Cited by 54 - Related Articles - Cached - Web Search - BL Direct

Whole program path profiling - group of 2 »

JR Larus, CW Fraser - US Patent 6,327,699, 2001 - Google Patents

... or program flow to enhance instruction **caching** and execution ... Apath **profiling** tool

is used to **instrument** the program ... as uses of the path to

identify **hot** subpaths ...

Cited by 10 - Related Articles - Web Search

Hot cold optimization of large Windows/NT applications - group of 10 »

R Cohn, PG Lowney - Microarchitecture, 1996. MICRO-29.

Proceedings of the 29th ..., 1996 - ieeexplore.ieee.org

... NT OM, that arranges code for instruction **cache**

performance ... Using **profile** information,

NT OM lays out code so ... routines into a frequently executed (**hot**) part and ...

Cited by 58 - Related Articles - Web Search - BL Direct

Cache-conscious structure definition - group of 20 »

TM Chilimbi, B Davidson, JR Larus - ACM SIGPLAN

Notices, 1999 - portal.acm.org

... In addition, BIT **instruments** the program to generate field access ... primary advantage

is the ability to pack more (**hot**) class instances in a **cache** block ...

Cited by 126 - Related Articles - Web Search - BL Direct

Profile-directed restructuring of operating system code - group of 6 »

WJ Schmidt, RR Roediger, CS Mestad, B Mendelson, I ... -

IBM Systems Journal, 1998 - research.ibm.com

... To **instrument** the fewest arcs, we then need to select ... basic blocks might be mapped

to instruction **cache** lines. Note that executing the **hot** trace when placed in ...

Cited by 27 - Related Articles - Cached - Web Search - BL Direct

Did you mean to search for: instrument **host** cache reduce profiling

Go o o o o o o o o o o g l e ►

Result Page: 1 2 3 4 5 6 7 8 9 10 Next

instrument hot cache reduce profiling 

[Google Home](#) - [About Google](#) - [About Google Scholar](#)

©2006 Google